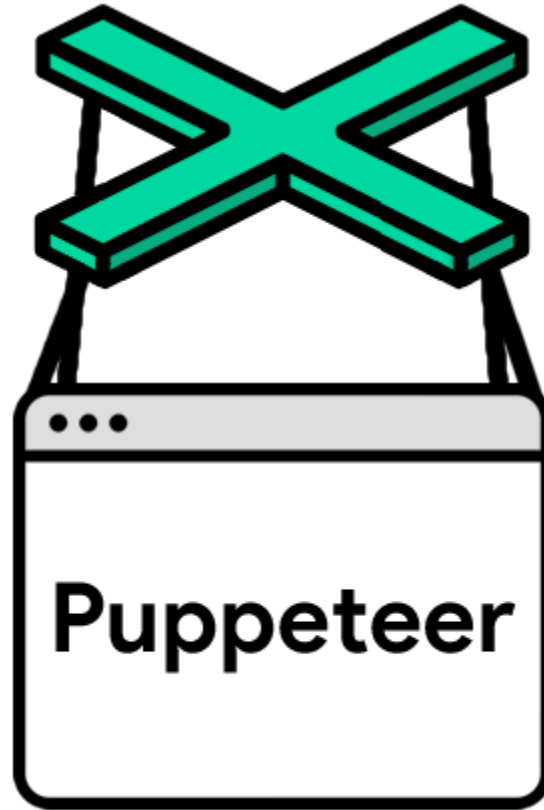




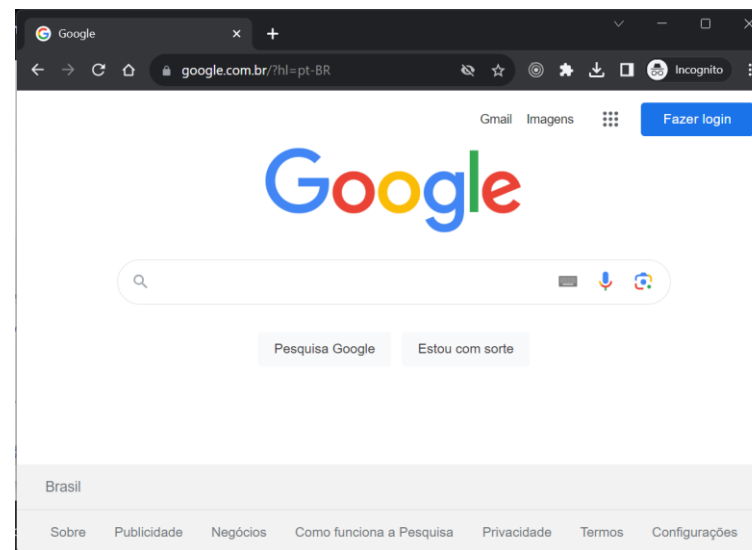
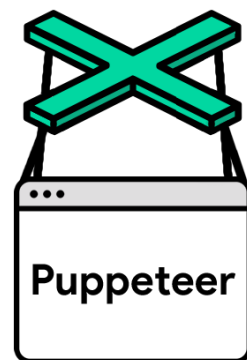
# **TREINAMENTO NODE.JS**

# Puppeteer



# Puppeteer

- O **Puppeteer** é uma biblioteca Node.js que fornece uma API de alto nível para controlar o Chrome/Chromium através do protocolo DevTools.

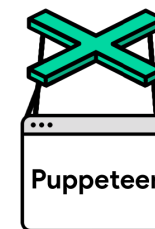


# Puppeteer



Ele é comumente usado para:

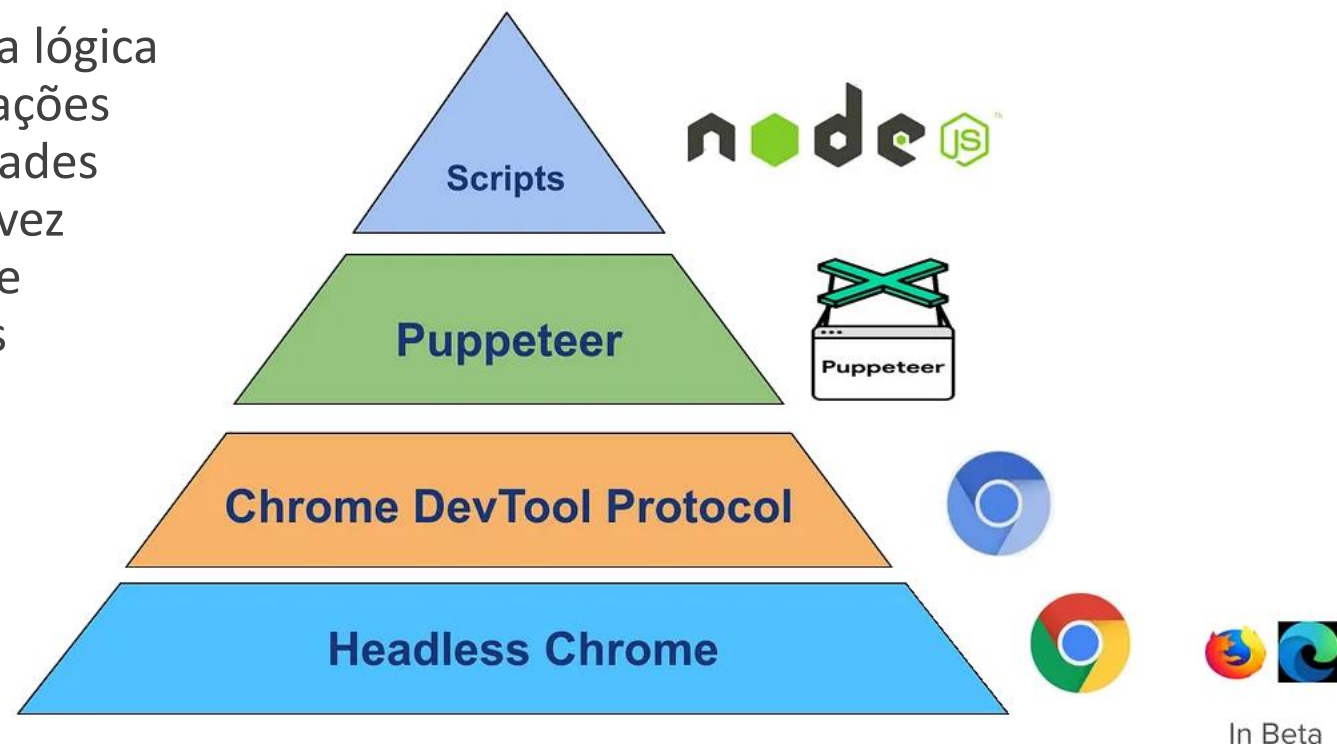
- **Navegação:** Abrir páginas da web, navegar por links, preencher formulários e realizar interações com elementos da página.
- **Captura de screenshots:** Puppeteer permite que você tire screenshots de páginas da web, o que é útil para testes visuais, geração de thumbnails, ou para capturar o estado de uma página em um determinado momento.
- **Geração de pdf:** Você pode gerar arquivos PDF a partir do conteúdo de páginas da web
- **Testes automatizados:** Puppeteer é frequentemente usado para escrever testes automatizados para aplicações web, permitindo verificar se uma aplicação se comporta corretamente em diferentes cenários de uso.
- **Web scraping:** É possível coletar dados de páginas da web, extrair informações e automatizar a coleta de dados para análise posterior..





# Puppeteer

- O Puppeteer permite que os desenvolvedores escrevam scripts em JavaScript para interagir com páginas da web de forma programática.
- Os scripts JavaScript são onde você cria a lógica personalizada para automatizar as interações com o navegador, usando as funcionalidades fornecidas pelo Puppeteer, que por sua vez se baseia no Chrome DevTools Protocol e no Headless Chrome para alcançar essas interações automatizadas.



# Puppeteer

- **Chrome DevTools** é um conjunto de ferramentas para desenvolvedores web integradas diretamente no navegador Google Chrome.



Chrome DevTools

[https://developer.chrome.com/docs/devtools/?utm\\_source=devtools](https://developer.chrome.com/docs/devtools/?utm_source=devtools)



# Puppeteer


- Atalhos para acessar o DevTools

- Pressione um atalho no Chrome dependendo do seu sistema operacional:

SO	Elementos	Console	Seu último painel
Windows or Linux	Ctrl + Shift + <b>C</b>	Ctrl + Shift + <b>J</b>	F12 Ctrl + Shift + <b>I</b>
Mac	Cmd + Option + <b>C</b>	Cmd + Option + <b>J</b>	Fn + F12 Cmd + Option + <b>I</b>



Chrome DevTools

- Esta é uma maneira fácil de memorizar os atalhos:
  - C** significa CSS (abre o painel Elementos  ).
  - J** para JavaScript.
  - I** sua escolha.



# Puppeteer

- Documentação

<https://pptr.dev/>

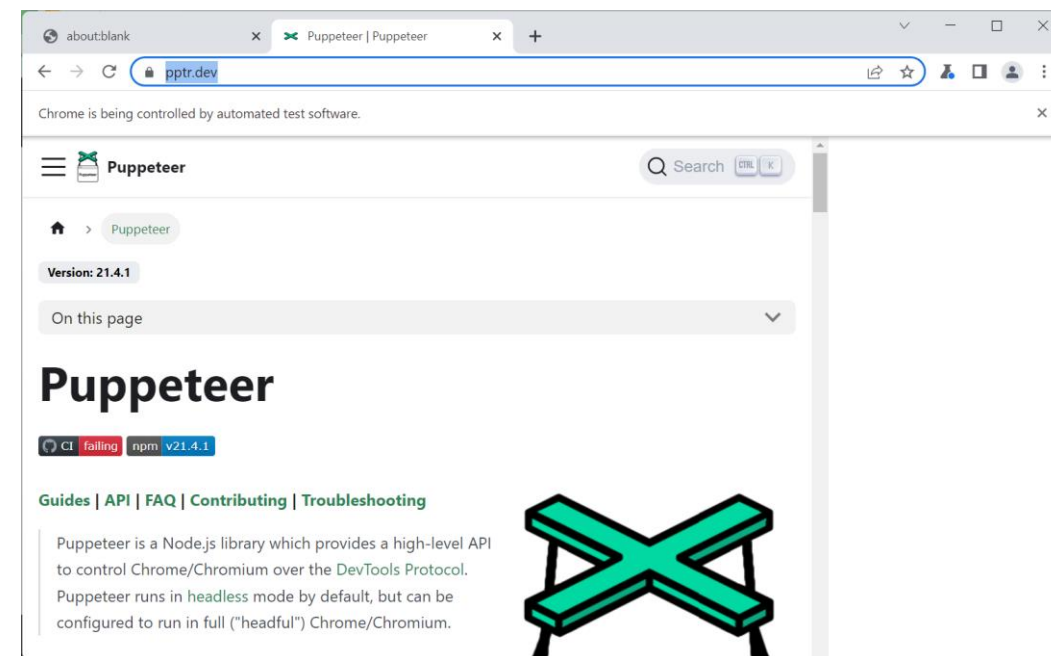
- Instalação

```
npm i puppeteer  
# or using yarn  
yarn add puppeteer  
# or using pnpm  
pnpm i puppeteer
```



# Puppeteer

- Ao instalar o puppeteer, automaticamente é baixado uma versão do **Chromium**.
- O **Chromium** é a versão de código aberto do navegador, enquanto o Google Chrome é a versão comercializada e mais conhecida, desenvolvida pelo Google, com algumas funcionalidades e componentes adicionais.
- O Chromium pode ser livremente baixado, modificado e distribuído por qualquer pessoa sob a licença de código aberto BSD.



# Puppeteer

- Utilização

```
console.log('Meu primeiro Robô em Puppeteer');

const puppeteer = require('puppeteer');

async function robo() {
  const browser = await puppeteer.launch({ headless: false });
  const page = await browser.newPage();
  await page.goto('https://example.com');
  await page.screenshot({ path: 'example.png' });

  await browser.close();
}

robo();
```

# Puppeteer

- Utilização

```
console.log('Meu primeiro Robô em Puppeteer');

const puppeteer = require('puppeteer');

async function robo() {
  const browser = await puppeteer.launch({ headless: false });
  const page = await browser.newPage();
  await page.goto('https://example.com');
  await page.screenshot({ path: 'example.png' });

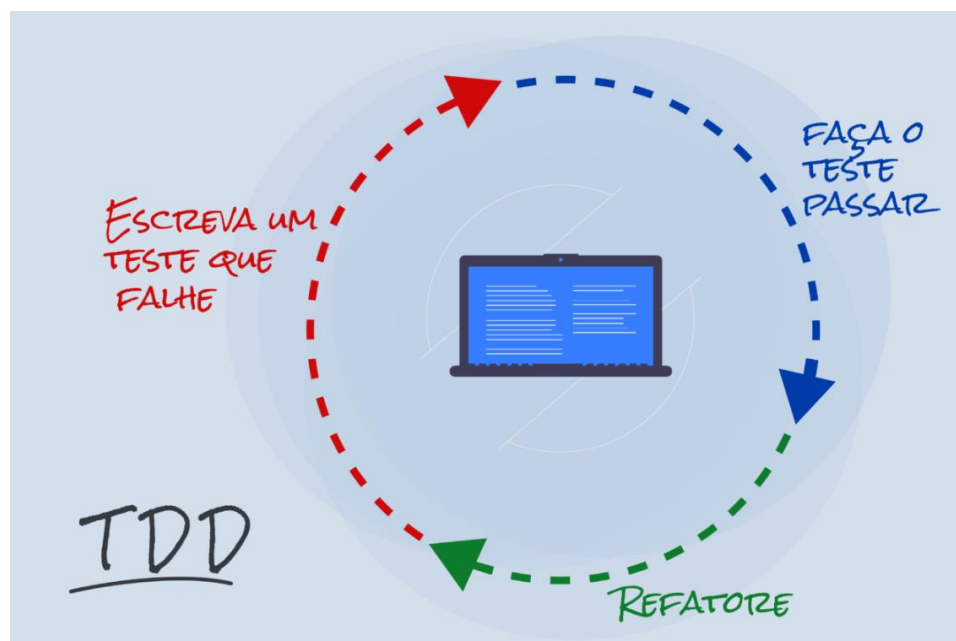
  await browser.close();
}

robo();
```

Por padrão, o Puppeteer não vai abrir janela visualmente (headless: true)

## TDD, ou Test-Driven Development (Desenvolvimento Orientado a Testes)

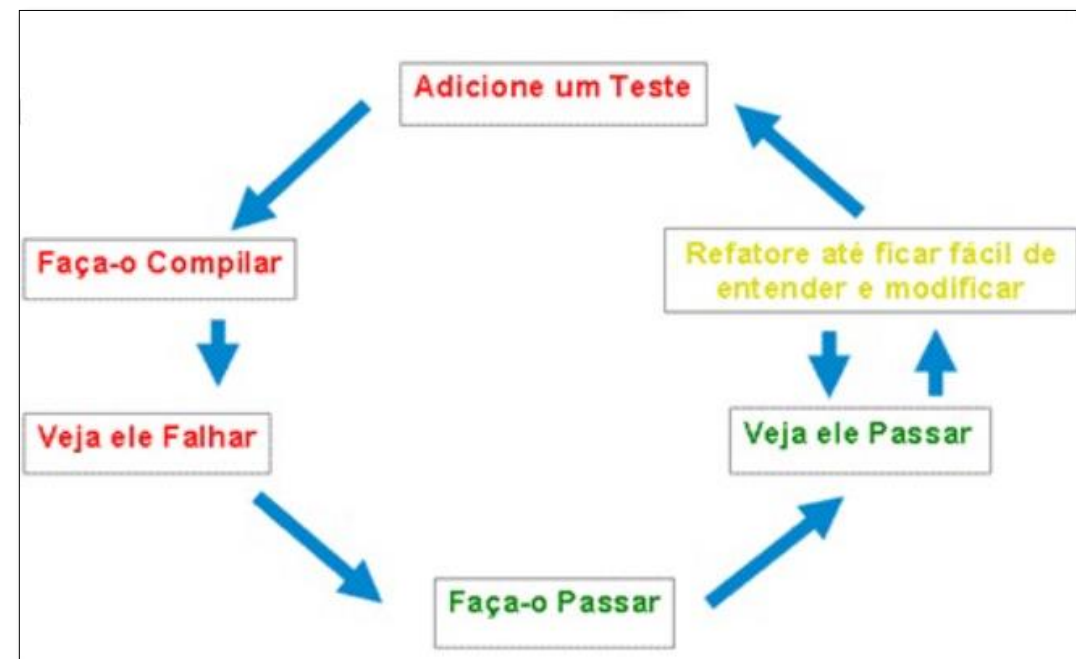
- É uma abordagem de desenvolvimento de software que enfatiza a criação de testes antes da escrita do código de produção.
- Essa prática é comum em muitos ambientes de desenvolvimento, incluindo o desenvolvimento Node.js.



# TDD

## TDD, ou Test-Driven Development (Desenvolvimento Orientado a Testes)

- Escrever um teste
- Codificar o mínimo possível para o teste passar
- Refatorar



## TDD na prática

### 1. Configurações iniciais

- Crie um novo diretório para o projeto (calculadoraApi)
- Entre no diretório calculadoraApi e execute no terminal “**npm init -y**” para criar o arquivo package.json padrão para gerenciar suas dependências.
- Instale as dependências do projeto: **npm install express mocha chai supertest --save-dev**
  - A estrutura do projeto será essa:

```
calculadoraApi/  
|-- src/  
|   |-- app.js  
|-- test/  
|   |-- api.test.js  
|-- package.json
```



## Pacotes:

- **mocha**: é um framework de teste para Node.js, amplamente utilizado para escrever testes unitários e de integração.
  - Ele permite definir *suites* de teste, escrever testes, fornecer ganchos (como `beforeEach`, `afterEach`, `describe`, `it`, etc.) e relatar os resultados dos testes.
- **chai**: Chai é uma biblioteca de asserções (ou assertions) para Node.js e navegadores. É frequentemente usado em conjunto com Mocha para realizar asserções em seus testes. Com Chai, você pode usar estilos de asserções como **expect**, **should** e **assert** para verificar se os resultados são os esperados.
- **supertest**: é uma biblioteca de teste de **HTTP** para Node.js, que fornece uma API simples e fluente para fazer solicitações HTTP a aplicativos Express (ou outros aplicativos HTTP). É usado para realizar testes de integração em APIs. Faz o request.

## TDD na prática

### 1. Escrevendo o primeiro teste

- Crie **app.teste.js** dentro da pasta `./test/`
- Escrever um teste para verificar se o servidor Express está respondendo com status 200 (OK).

```
JS app.teste.js X
test > JS app.teste.js > ...
1  const request = require('supertest');
2  const app = require('../src/app');
3  const { expect } = require('chai');
4
5  describe('API Tests', () => {
6    it('should return 200 OK', (done) => {
7      request(app)
8        .get('/')
9        .expect(200, done);
10   });
11 });
12
```

# TDD



## TDD na prática

### 1. Executar primeiro teste

- No terminal, execute: **npx mocha**
- Deve retornar erro, pois o arquivo ainda não existe

# TDD



Os testes são executados via linha de comando, com:

`npx mocha`

Ou

`./node_modules/.bin/mocha`

- O **npx** é uma ferramenta que vem junto com o Node.js, introduzida na versão 5.2.0 e superior.
- Ela é projetada para ajudar a executar pacotes Node.js de forma mais conveniente, especialmente quando você deseja executar pacotes que não estão instalados globalmente na sua máquina.

## TDD na prática

### 1. Escrevendo a primeira rota da API

- Crie app.js dentro da pasta src
- Escrever primeira rota.

```
JS app.js ×
src > JS app.js > ...
1  const express = require('express');
2  const app = express();
3
4  app.get('/', (req, res) => {
5    res.status(200).json({ message: 'Hello, world!' });
6  });
7
8  module.exports = app;
9
```

## TDD na prática

### 1. Executar primeiro teste

- No terminal, execute: `npx mocha`

```
PS C:\aula7\Testes\calculadoraApi> npx mocha
```

```
API Tests  
  ✓ should return 200 OK
```

```
1 passing (34ms)
```



# Atividade



- Escrever dois casos de teste na **calculadoraApi** (dentro da pasta /aula7/Testes/):
  - “Deve retornar 5 ao dividir 15/3”
  - “Deve retornar erro ao dividir 15/0”
- Executar o teste, com **npx mocha**, e enviar evidências do resultado do teste
- Atualmente esses dois testes falham porque não foram implementados:

## API Tests

✓ should return 200 OK

## Testando operações

- ✓ Deve retornar 5 ao subtrair 8 - 3
- ✓ Deve retornar 5 ao somar 2 + 3
- 1) Deve retornar 5 ao dividir 15 / 3
- 2) Deve retornar erro ao dividir 15 / 0
- ✓ Deve retornar erro ao multiplicar 5 \* 1

4 passing (4s)

2 failing

```
JS app.test.js x
Testes > calculadoraApi > test > JS app.test.js > ...
35   });
36
37   it('Deve retornar 5 ao dividir 15 / 3', (done) => {
38     // Escrever código aqui
39   });
40
41   it('Deve retornar erro ao dividir 15 / 0', (done) => {
42     // Escrever código aqui
43   });
44
45   it('Deve retornar erro ao multiplicar 5 * 1', (done) => {
46     request(app)
47       .get('/divisao/15/3')
48       .expect(200)
49       .end((err, res) => {
50         if (err) return done(err);
51         expect(res.body.result).to.equal(5);
52         done();
53       });
54   });
55
56 });
57
```